Lab 5: Digital Audio

Introduction

In this lab you will use your MCU to play music by using timers to generate square waves by toggling a GPIO pin at a specific frequency for specified durations.

Learning Objectives

By the end of this lab you will have...

- Built a circuit to enable an I/O pin from your MCU to drive a speaker
- Implemented the timer functionality available on the MCU by reading the datasheet and writing your own library in C from scratch

Requirements

Build a system to play music on a speaker. Use your MCU, an LM386 audio amplifier, and an 8-ohm speaker. The MCU should read a list of notes specifying the pitch (in Hz) and duration (in ms) of each note. It should generate a corresponding sequence of square waves. A frequency of 0 indicates a rest (silence for the given duration). A duration of 0 indicates the end of the song. Your system should play accurate pitches regardless of the frequency. Test your system on the score of Für Elise, which is provided. Your code should manually define `#define` macros and register structures for the memory-mapped registers you need to manipulate to help you practice developing your own device drivers from scratch using only the reference manual as a reference. In particular, this means that you are not allowed to use the CMSIS headers for your device.

Resources

- Starter Code from GitHub Repo

Lab 5 Specifications

Lab-specific Specifications

Proficiency

- ☐ Design plays *Für Elise* from provided starter code
- ☐ Note durations match the durations specified in the starter code for *Für Elise* (i.e., the tune plays at the correct tempo)
- ☐ Individual pitches are accurate to within 1% across the frequency range (of 220-1000 Hz) (calculations should be provided in the report to verify this)
- ☐ All rests (pauses with no sound) are played properly
- ☐ Code uses `#define` macros for memory-mapped registers

Excellence

- ☐ Report contains accurate calculations for **minimum duration** supported
- ☐ Report contains accurate calculations for **maximum duration** supported
- ☐ Report contains accurate calculations for **minimum frequency** supported
- ☐ Report contains accurate calculations for **maximum frequency** supported
- ☐ Report provides documentation and calculations to show that the durations and pitches are correct based on the timer configuration.
- ☐ Design contains potentiometer to control the output volume.
- ☐ Design plays an extra composition of your choice. You need not compose the tune from scratch, it is acceptable to transpose an existing tune.

General Specifications

Proficiency

General Schematic Specifications

- ☐ All pin names labeled
- ☐ All pin numbers labeled
- ☐ Crossing wires clearly identified as junction or unconnected
- ☐ Neat layout (e.g., clear organization and spacing)
- ☐ All parts labeled with part number
- ☐ All component values present

Block Diagram

- ☐ Block diagram present with one block per SystemVerilog module
- ☐ Each block includes all input and output signals

HDL & Code Specifications

General Formatting

- ☐ Descriptive filename (e.g., `lab2_jb.sv`)
- ☐ Descriptive variable names
- ☐ Neat formatting (e.g., standard indentation, consistent formatting for variable names (kebab-case/snake_case/camelCase/PascalCase))
- ☐ Descriptive and clear function/module names

Comments

- ☐ Comments to indicate the purpose of each function/module

Lab Writeup/Summary

- ☐ Brief (e.g., 3-5 sentence) description of the main goals of the assignment and what was done.
- ☐ Explanation of design approach. How did you go about designing and implementing the design?
- ☐ Explanation of testing approach. How did you verify your design was behaving as expected?
- ☐ Statement of whether the design meets all the requirements. If not, list the shortcomings.
- ☐ Number of hours spent working on the lab are included.
- ☐ Writeup contains minimal spelling or grammar issues and any errors do not significantly detract from clarity of the writeup.
- ☐ (Optional) List commments or suggestions on what was particularly good about the assignment or what you think needs to change in future versions.

Excellence

General Schematic Specifications

- ☐ Standard symbols used for all components where applicable
- ☐ Signals "flow" from left to right where possible (e.g., inputs on left hand side, outputs on right hand side)
- ☐ Title block with author name, title, and date

HDL & Code Specifications

General Formatting

- ☐ Name, email, and date at the top of every file
- ☐ Comment at the top of each source code file to describe what is in it
- ☐ Clear and organized hierarchy (e.g., deliniation between top level modules and submodules)

Testbenches

- ☐ Testbenches written for each individual module to demonstrate proper operation
- ☐ Testbench output included in the report

Lab Writeup/Summary

- ☐ Writeup is free of spelling and grammar issues

Open specifications in new tab.

Discussion

You can find `lab5_starter.c` on the class web page with the Für Elise score provided as an array of ordered pairs of pitch frequencies and durations.

A goal of this lab is for you to learn to interpret a datasheet and figure out how the timer works. Write your own code from scratch to use the system timer.

The GPIO pins don't generate enough output current to play satisfactory music directly on the speaker, so use an LM386 audio amplifier between the MCU and the speaker. Do not connect the MCU directly to the speaker, as the current draw could damage it. The datasheet shows AC coupling from the amplifier to the speaker, but you can leave out the capacitors and resistors and produce an acceptable square wave. Volume control is optional (but recommended for your own sanity and that of your roommates and labmates). There are only a limited number of speakers available in the lab so please leave the speakers in the supply cabinet when you leave the lab. Do not leave them attached to your breadboard when you are done working. If you kill a speaker, throw it away rather than putting it back in the cabinet for your unfortunate classmates.

Optional Exercise

In the past students have enjoyed composing their own tune to play on their MCU. The following information may help if you wish to compose your own piece of music.

The duration depends on an arbitrary choice of tempo (speed at which the piece is played). If a whole note is chosen to be 1/2 a second long, other notes follow accordingly:

Note Type	Duration [s]
Whole	0.5
Half	0.25
Quarter	0.125
Eighth	0.0625
Sixteenth	0.03125

Recall that the A above middle C (called A4) is 440 Hz (at least in the United States) and that an octave spans a factor of 2 in frequency. There are twelve notes in an octave spaced evenly on a geometric scale, so each is separated in frequency by a factor of 21/12.

Note	Frequency (Hz)
A3	220
A3 sharp / B3 flat	233.1
B3	246.9
C4 (middle C)	261.6
C4 sharp / D4 flat	277.2
D4	293.7
D4 sharp / E4 flat	311.1
E4	329.6
F4	349.2
F4 sharp / G4 flat	370.0
G4	392.0
G4 sharp / A4 flat	415.3
A4	440
A4 sharp / B4 flat	466.2
B4	493.9
C5	523.3
C5 sharp / D5 flat	554.4
D5	587.3
D5 sharp / E5 flat	622.2
E5	659.2
F5	698.4

Note	Frequency (Hz)
F5 sharp / G5 flat	740.9
G5	784.0
G5 sharp / A5 flat	830.6
A5	880

What to Turn In

When you are done, have your lab checked off by the instructor. You should thoroughly understand how it works and what would happen if any changes were made. Turn in your lab writeup including the following information:

- Schematics of the breadboarded circuit.
- Your source code.
- How many hours did you spend on this lab? This will not count toward your grade.

Credits

This lab was originally developed in 2015 by Alex Alves '16, redesigned for the Mudd Mark 5.1 in 2019 by Caleb Norfleet '21, and redesigned for the Mudd MkVI in Fall 2021 by Prof. Josh Brake.